

Europäisches Patentamt

European Patent Office

Office européen des brevets

(11)

EP 1 051 028 A1

(12)

## EUROPEAN PATENT APPLICATION

(43) Date of publication:

08.11.2000 Bulletin 2000/45

(51) Int Cl. 7: H04N 5/00

(21) Application number: 99108916.0

(22) Date of filing: 05.05.1999

(84) Designated Contracting States:

AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU  
MC NL PT SE

Designated Extension States:

AL LT LV MK RO SI

- Tiedemann, Stephen,  
c/o Sony Int. (Europe) GmbH  
70736 Fellbach (DE)
- Schwager, Andreas,  
c/o Sony Int. (Europe) GmbH  
70736 Fellbach (DE)

(71) Applicant: Sony International (Europe) GmbH  
10785 Berlin (DE)

(74) Representative:

MÜLLER & HOFFMANN Patentanwälte  
Innere Wiener Strasse 17  
81667 München (DE)

(72) Inventors:

- Szucs, Paul, c/o Sony Int. (Europe) GmbH  
70736 Fellbach (DE)

## (54) Control module of a tuner for a HAVI network

(57) A tuner component module within a home network based on the HAVI standard, i.e. a tuner FCM, according to the present invention includes the functionality to provide the capability of the tuner and/or to introduce the DVB/DTV concepts of components and events to the HAVI network. In this context components could be audio, video or data streams associated with a service and events could e. g. be information for an electron-

ic programming guide (EPG). The capability of a tuner in this context are all functions supported by the tuner. Therefore, a tuner FCM (1) according to the present invention includes at least one of the following APIs and the corresponding (data/signal) structures necessary to have a proper functionality: GetItemComponentList API (2); GetEventList API (3), SelectItemComponent API (4), GetCurrentItem-Component API (5) and GetCapability API (6).

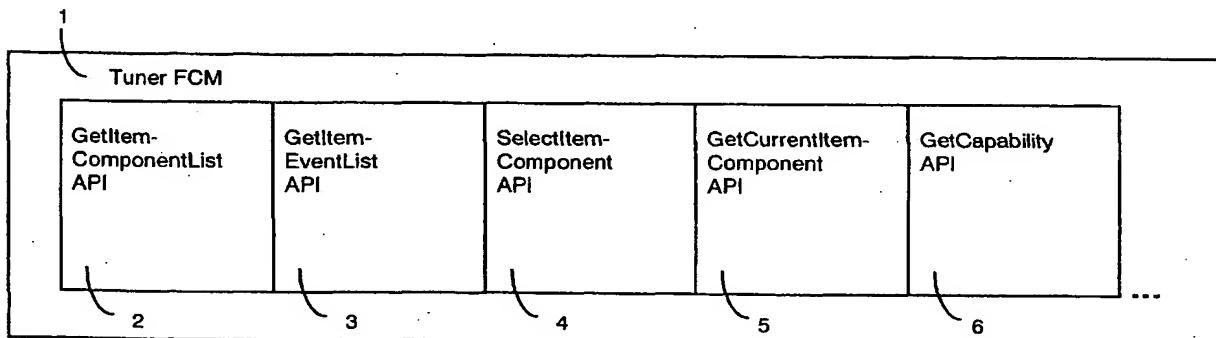


Fig. 1

**Description**

[0001] This invention relates to a tuner component module within a home network, in particular within a home network based on the HAVi standard, which is adapted to support the access of transmitted items. A tuner controlled via a network is accessed in a HAVi network via its tuner component module (FCM) which presents one or more application programming interfaces (APIs) to other elements within the home network.

5 [0002] Within the HAVi specification, e.g. in the version HAViOBn1 of December 10, 1998, several APIs are defined for a tuner functional component module (tuner FCM). These APIs allow only a very simple control of a tuner device within the home network. Basically, the tuner device can only provide unnamed and unspecified lists of items, i.e. services or channels, which can be selected via the network. A selected item will then be received and distributed 10 within the network by the tuner device.

15 [0003] In the above identified version of the HAVi specification an item is referred to by a descriptive name thereof. The meaning of this value is dependent on the tuner. An item could e. g. be the channel number, the service name or the frequency for a certain service. Each item is uniquely identified by a predefined index number. A GetItemList API returns a list of items and indexes available from the tuner. A SelectItem API allows the selection of an item which is identified by its index number and causes the output of said selected item by the tuner to a specified plug. Output in this sense means to append or remove the selected item to/from the data stream to this specified plug. A GetCurrentItem API returns a list of items and indexes currently being output to a specified plug and an ItemChanged API notifies a change in items of a specified plug. Furtheron, a tuner notification attribute itemSelection is defined which provides 20 new settings of item selections on the tuner output plug.

[0004] Therefore, the object of the present invention to provide a tuner component module providing an improved control of a tuner in the home network. This object is solved by a tuner component module according to claim 1 which has means to access at least one component and/or event transmitted within one item and/or to show its capability.

25 [0005] Therefore, according to the present invention a tuner component module adapted to the HAVi standard, i.e. a tuner FCM, includes the functionality to provide its capability and/or to introduce the DVB/DTV concepts of components and events to the HAVi network. In this context components could be audio, video or data streams associated with a service and events could e. g. be information for an electronic programming guide (EPG). The capability of the tuner component module in this context are all supported functions.

30 [0006] Preferred embodiments of the present invention are defined in the subclaims 2 to 11 which define APIs for the access of components and events as well as a structure providing the kind of an item, such as TV, radio or data.

[0007] In the following a descriptive example of a tuner component module according to the present invention will be given. Such a tuner FCM can be realized in hardware as described in connection with the first preferred embodiment of the present invention shown in Fig. 1 or in software as described in connection with the second preferred embodiment of the present invention.

35 [0008] To be able to specify an item besides its name or channel/frequency the tuner FCM 1 according to the first preferred embodiment of the present invention which is shown in Fig. 1 defines an item type additionally to the structure, the APIs, the events and notifications defined in a tuner FCM according to the above indicated version of the HAVi standard. Such a type could e.g. indicate that a certain item is used for an audio service, an audio/video service or a data service. Furtheron, a type ensemble can refer to a collection of items, generally meaning the complete multiplex 40 or channel the receiver being tuned to. Such a categorisation of items allows a more specified access.

[0009] Furtheron, the tuner component module 1 according to the present invention introduces the concepts of components and events known from DVB and DTV to HAVi. Therefore, it defines a component type that allows the definition of certain components within one item, such as a video component, an audio component, a subtitle component, a teletext component, or a data component. Of course, one item can comprise several components of one type. Additionally an index is provided to uniquely identify a component.

45 [0010] Both, components and events for a particular item available from the tuner, e. g. the item currently being output by the tuner, can be collected in a respective list which can be requested similar as the item list already specified within the above identified version of the HAVi specification. Therefore, the tuner FCM 1 according to the present invention includes a GetItem-ComponentList API 2 to collect and output a list of components and corresponding indexes of a particular item available from the tuner and a GetItemEventList API 3 to collect and output a list of events carried by a particular item available from the tuner.

50 [0011] Every component of an item can be selected with the help of its index via a SelectItemComponent API 4. A selected item component is then caused to be output by the tuner. Of course, not only one component of one item can be selected at a time, but several can be selected simultaneously, as it would be the case of reception and distribution of a video stream as e.g. first component of an item 'ARD' and a first audio stream as e.g. third component 'original language' corresponding to said video stream. The second component which is received by the tuner, but not distributed within the network, since it is not selected could e.g. be an audio stream of the 'german translation' corresponding to the video stream.

[0012] A GetCurrentItem API 5 returns a list of components and indexes of one or more item(s) currently being output by the tuner device to a specified plug.

[0013] Furtheron, besides the possibility to access components of an item with the help of the component and/or index also the access of events with the help of an event index is possible according to the present invention. In this case access includes not the selection of an event, since an event is used for information purposes only. Therefore, only a list of events carried by a particular item available from the tuner will be returned from the GetItemEventList API 3 as described above.

[0014] A further advantage of the present invention is the provision of the tuner FCM capability which can be requested via a GetCapability API 6 and which is output in a list similar to e. g. the VCR FCM capability already defined in chapter 6.2 of the above identified version of the HAVi specification.

[0015] In the following the tuner services, tuner data structures, tuner APIs, tuner events and tuner notification attributes are shown as they could be specified in chapter 6.1 of a further version of the HAVi specification or in a software version of a tuner FCM according to a second preferred embodiment of the present invention. The tuner services, tuner data structures, tuner APIs and tuner notification attributes which are new in comparision with chapter 6.1 of the above identified version of the HAVi specification are indicated by a \*-sign.

20

25

30

35

40

45

50

55

## 6.1 Tuner FCM

### 6.1.1 Tuner Services

Service	Comm Type	Locality	Access	Resv Prot
Tuner::GetItemList	M	global	all	
Tuner::GetItemComponentList *	M	global	all	
Tuner::GetItemEventList *	M	global	all	
Tuner::SelectItem	M	global	all	yes
Tuner::SelectItemComponent *	M	global	all	yes
Tuner::GetDownList	M	global	all	
Tuner::SelectData	M	global	all	yes
Tuner::GetCurrentItem	M	global	all	
Tuner::GetCurrentItemComponent *	M	global	all	
Tuner::GetCurrentData	M	global	all	
Tuner::GetCapability *	M	global	all	
ItemChanged	E	global	Tuner (all)	

### 6.1.2 Tuner Data Structures

#### ItemType enumerator \*

```
enum ItemType{    ITEM_AVSERVICE,
                  ITEM_AUDIOSERVICE,
                  ITEM_DATASERVICE,
                  ITEM_ENSEMBLE,
                  ITEM_UNSPECIFIED,
};
```

#### ItemIndex

```
struct ItemIndex {
    ushort          Index;
    ItemType        itemType;
    String          Item;
};
```

#### Element

#### Index

#### Description

The number of the index to uniquely identify the Item below.

**ItemType**

Categorization of item. There are three types of item defined for services. The type Ensemble can refer to a collection of items, generally meaning the complete multiplex or channel being tuned to.

**Item**

A descriptive name of the item.

**ItemIndexList**

```
typedef sequence<ItemIndex> ItemIndexList;
```

**ComponentType enumerator \***

```
enum ComponentType{ COMPONENT_VIDEO,
COMPONENT_AUDIO,
COMPONENT_SUBTITLE,
COMPONENT_TELETEXT,
COMPONENT_DATA,
COMPONENT_UNSPECIFIED,
};
```

**ComponentIndex \***

```
struct ComponentIndex {
    ushort Index;
    ComponentType componentType;
    String Component;
};
```

**Element****Index****Description**

The number of the index to uniquely identify the Component below.

**componentType**

Categorization of component

**Component**

A descriptive name of the component.

**ItemComponentIndexList \***

```
typedef sequence<ComponentIndex> ItemComponentIndexList;
```

**EventIndex \***

```
struct EventIndex {
    ushort Index;
    String EventName;
    DateTime StartTime;
    DateTime Duration;
    String EventDescription;
};
```

<b>Element</b>	<b>Description</b>
Index	The number of the index to uniquely identify the Event below within the service.
5 EventName	A descriptive name of the event.
StartTime	Start date & time of event
EndTime	End date & time of event
10 EventDescription	Text containing extended description

**ItemEventIndexList \***

```
15     typedef sequence<EventIndex> ItemEventIndexList;
```

**DataIndex**

```
20     struct DataIndex {
              ushort          Index;
              String          Data;
      };
```

<b>Element</b>	<b>Description</b>
Index	The number of the index to uniquely identify the Data below.
25 Data	A descriptive name of the data, which can be the data name or the PID number and so on.
30	The meaning of this value is dependent on the functional component (Tuner).

**DataIndexList**

```
35     typedef sequence<DataIndex> DataIndexList;
```

**TunerCapability enumerator \***

```
40     enum TunerCapability { CAP_GetItemComponentList,
                           CAP_GetItemEventListCurrent,
                           CAP_GetItemEventListCurrentNext,
                           CAP_GetItemEventListDay,
                           CAP_GetItemEventListWeek,
                           CAP_GetItemEventListAll,
                           CAP_SelectItemComponent,
      };
```

**TunerCapabilityList \***

```
50     typedef sequence<TunerCapability> TunerCapabilityList;
```

### 6.1.3 Tuner API

#### Tuner::GetItemList

##### Prototype

```
10 Status Tuner::GetItemList(
    in uchar listNumber,
    out ItemIndexList itemIndexList)
```

##### Parameters

- 15 • **listNumber** : Specifies the list number of the index list to be returned.
- **itemIndexList** : A list of items and indexes. The definitions of this **ItemIndexList** list and the **ItemIndex** structure are described above.

##### Description

20 This API returns a list of items and indexes, in accordance with the **ListNumber**. These indexes within the list can be used when issuing the **Tuner::SelectItem** API.

The Tuner may hold several lists internally. The usage of lists is dependant on the Tuner. The example of **listNumber** and its relationships with the items are illustrated below.

25 Ex1) Item is used as (preset or preferred) channel number.

The preset channel numbers are area (and/or broadcast system) dependant, and independent of the tuner. Thus, these numbers are the same between each tuner in the same area (and/or broadcast system).

30 In most case, the preferred channel numbers are a collection of user preference items. Thus, these numbers may not be the same between each tuner.

Tuner type	Item format	Meaning (example)
all types	character string	„888“ = 888 CH

40 Ex2) Item is used as (preset or preferred) item name (service name).

Tuner type	Item format	Meaning (example)
all types	Character string	„BBC“ = BBC service

45 Ex3) Item is used as frequency.

Tuner type	Item format	Meaning (example)
Analog Audio (AM)	Character string	„1234“ = 1234 kHz
Analog Audio (FM)	Character string	„123.4“ = 123.4 MHz

Ex4) item is used as VCH.

Tuner type	Item format	Meaning (example)
ATSC (US-DTV)	Area(3digits) + Major CH# (3digits) + Minor CH# (3digits)	"001023004" = Area number : 1 Major CH# : 23 Minor CH# : 4

#### Error codes

- `Tuner::ENOT_SUPPORTED` – if the target device does not support the specified list number.

### Tuner::SelectItem

**Prototype**  
`Status Tuner::SelectItem(  
 in short index, in short plug, in short action)`

#### Parameters

- `index` : Specifies the index number of the item to be selected.
- `plug` : Number of plug to output the above item to.
- `action` : Specifies the action to be carried out on the output signal. The following values can be set.

Definition	Value and Meaning
APPEND	0x0000: Add (multiplex) the specified item to the output signal.
REPLACE	0x0001: Remove all current items from the output signal, and output the specified item.
REMOVE	0x0002: Remove the specified item from the output signal.
CLEAR	0x0003: Stop the output of all items on the specified plug. In this case, the <code>Index</code> parameter may be a NULL value.

#### Description

This API selects the items, which are identified by `Index`, and outputs (appends or removes, etc) it to the specified plug.

#### Error codes

- `Tuner::EREJECTED` – if the REMOVE/REPLACE/CLEAR action is specified when the item is not present on the specified plug.

**Tuner::GetItemComponentList \*****Prototype**

```
5 Status Tuner::GetItemComponentList(
    in short itemIndex,
    out ItemComponentIndexList itemComponentIndexList)
```

**Parameters**

itemIndex	: item index for which component info is required.
itemComponentIndexList	: A list of item components and indexes.

**Description**

An item may consist of several components, for example a video stream and several audio streams for multiple languages. This API returns a list of components comprising all components of a particular item available from the tuner. These indexes within the list can be used when issuing the Tuner::SelectItemComponent API.

**Tuner::SelectItemComponent \*****Prototype**

```
25 Status Tuner::SelectItemComponent(
    in short itemIndex, in short itemComponentIndex, in short
    plug,
    in short action)
```

**Parameters**

• itemIndex	: Specifies the index of the item for which the component is to be selected.
• itemComponentIndex	: Specifies the index of the component to be selected.
• plug	: Number of plug to output the above item to.
• action	: Specifies the action to be carried out on the output signal.

The following values can be set.

Definition	Value and Meaning
APPEND	0x0000: Add (multiplex) the specified component to the output signal.
REPLACE	0x0001: Remove all current components from the output signal, and output the specified item.
REMOVE	0x0002: Remove the specified component from the output signal.

**Description**

This API selects the item component identified by Index, and outputs (appends or removes, etc) it to the specified plug. The relevant item must already be selected.

**Error codes**

- **Tuner::EREJECTED** – if the REMOVE/REPLACE/CLEAR action is specified when the item is not present on the specified plug.

5

**Tuner::GetItemEventList \***

10

```
enum EventInfoLevel{ EV_CURRENT,
                     EV_CURRENTNEXT,
                     EV_TODAY,
                     EV_WEEK,
                     EV_ALL,
};
```

15

**Prototype**

```
Status Tuner::GetItemEventList(
    in short itemIndex, in EventInfoLevel eventInfoLevel,
    out ItemEventIndexList itemEventIndexList)
```

20

**Parameters**

- **itemIndex** : service index for which event info is required.
- **eventInfoLevel** : amount of event information required, as defined above.
- **itemIndex** : item index for which event info is required. Special reserved
- **itemEventIndexList** : A list of events and indexes. The definitions of this list and the **itemEventIndex** structure are described above.

25

**Description**

30

This API returns a list of events (content items) carried by a particular item available from the tuner. These lists are for information only; no selection is possible on an event.

**Tuner::GetDataList**

35

**Prototype**

```
Status Tuner::GetDataList(
    out DataIndexList dataIndexList)
```

40

**Parameters**

- **dataIndexList** : A list of data (such as PID) and indexes. The definitions of this **DataIndexList** list and the **DataIndex** structure are described above.

45

**Description**

This API returns a list of data (strings – such as a data name or PID number) and indexes. These indexes within the list can be used when issuing the **Tuner::SelectData** API.

50

**Tuner::SelectData**

55

**Prototype**

```
Status Tuner::SelectData(
    in ushort index, in short plug,
    in short action, in short flow)
```

**Parameters**

- **index** : Specifies the index number of the data to be selected.
- **plug** : Number of plug to output the above data to.
- **action** : Specifies the action to be carry out on the output signal. The values allowed are the same as for the **SelectItem** API.
- **flow** : How the data should be transmitted. The following values can be set.

<b>Definition</b>	<b>Value and Meaning</b>
SEND_ONCE	0x0000: send the data once.
SEND_NEW	0x0001: send each new version of the data.
SEND_ALL	0x0002: send all occurrences

**Description**

This API selects the data (such as NIT or PMT), which is identified by **index**, and outputs (append or remove, etc.) it to the specified plug. The **Flow** parameter specifies the output method.

**Error codes**

**Tuner::EREJECTED** if the REMOVE/REPLACE/CLEAR action is specified when the data is not present on the specified plug.

**Tuner::GetCurrentItem****Prototype**

```
30 Status Tuner::GetCurrentItem(
    in short plug, out ItemIndexList currentItemList)
```

**Parameters**

- **plug** : The number of the plug to return the current output of.
- **currentItemList** : A list of items and indexes currently being output. The definitions of this **ItemIndexList** list and the **ItemIndex** structure are described above.

**Description**

This API returns a list of items and indexes, currently being output to the specified plug.

**Tuner::GetCurrentItemComponent \*****Prototype**

```
45 Status Tuner::GetCurrentItemComponent(
    in short plug, in short itemIndex,
    out ItemComponentIndexList currentItemComponentList)
```

**Parameters**

- **plug** : The number of the plug for which to return the current output.
- **itemIndex** : The number of the item for which to return the current components.
- **currentItemComponentList** : A list of items and indexes currently being output. The definitions of this **ItemComponentIndexList** list and the **ItemComponentIndex** structure are described above.

**Description**

This API returns a list of components and indexes of the item currently being output to the specified plug.

**Tuner::GetCurrentData****Prototype**

```
Status Tuner::GetCurrentData(
    in short plug, out DataIndexList currentDataList)
```

**Parameters**

- **plug** : The number of the plug to return the current data output of.
- **currentDataList** : A list of data and indexes currently being output. The definitions of this **ItemDataList** list and the **DataIndex** structure are described above.

**Description**

This API returns a list of data and indexes, currently being output to the specified plug.

**Tuner::GetCapability \*****Prototype**

```
Status Tuner::GetCapability(
    out TunerCapabilityList capabilityList);
```

**Parameters**

- **capabilityList** : A list of capabilities supported by the Tuner Functional Component Module. The definitions of this **TunerCapabilityList** list and the **TunerCapability** enumerator are described above.

**Description**

This API returns the capabilities of the Tuner Functional Component Module.

**6.1.4 Tuner Events****ItemChanged****Prototype**

```
void ItemChanged(
    in short plug,
    in ItemIndexList currentItemList)
```

50

55

**Parameters**

- **plug**: The number of the plug, which notification of changes in its output are to be given.
- **currentItemList**: A list of items and indexes currently being output (after a change occurs). The definitions of this **ItemIndexList** list and the **ItemIndex** structure are described above.

**Description**

This API notifies a change in items of a plug indicated by the **plug** parameter.

**6.1.5 Tuner Notification Attributes****Tuner::itemSelection****Attribute**

```
struct {short, ItemIndexList} itemSelection
```

**Description**

New settings of item selections on outputplug. Only useful with comparator **ANY**.

**Tuner::itemComponentSelection \*****Attribute**

```
struct {short, short itemIndex, ItemComponentIndexList} itemComponentSelection
```

**Description**

New settings of item component selections on outputplug. Only useful with comparator **ANY**.

**Tuner::dataSelection****Attribute**

```
struct {short, DataIndexList} dataSelection
```

**Description**

New settings of data selections on outputplug. Only useful with comparator **ANY**.

**Claims**

1. Tuner component module (1) within a home network based on the HAVi standard adapted to support the access of transmitted items, **characterized by**  
means to access at least one component and/or event transmitted within one item and/or to show the capability of a connected tuner.
2. Tuner component module according to claim 1, **characterized in that** said means allows the generation of a list of components comprising all components of a particular item available from the tuner.

3. Tuner component module according to claim 2, **characterized in that** said means contains a GetItemComponentList API (2).
4. Tuner component module according to claim 2 or 3, **characterized in that** said means allows the generation of a list of components and indexes of an item currently being output by the tuner.
5. Tuner component module according to claim 4, **characterized in that** said means contains a GetCurrentItem-Component API (5).
10. 6. Tuner component module according to anyone of claims 1 to 5, **characterized in that** said means allows to select an item component and causes the output of said selected item component by the tuner.
7. Tuner component module according to claim 6, **characterized in that** said means contains a SelectItemComponent API (4).
15. 8. Tuner component module according to anyone of claims 1 to 7, **characterized in that** said means allows the generation of a list of events carried by a particular item available from the tuner.
9. Tuner component module according to claim 8, **characterized in that** said means contains a GetItemEventList API (3).
20. 10. Tuner component module according to anyone of claims 1 to 9, **characterized in that** said means contains a GetCapability API (6).
25. 11. Tuner component module according to anyone of claims 1 to 10, **characterized in that** an item can be specified by its type.

30

35

40

45

50

55

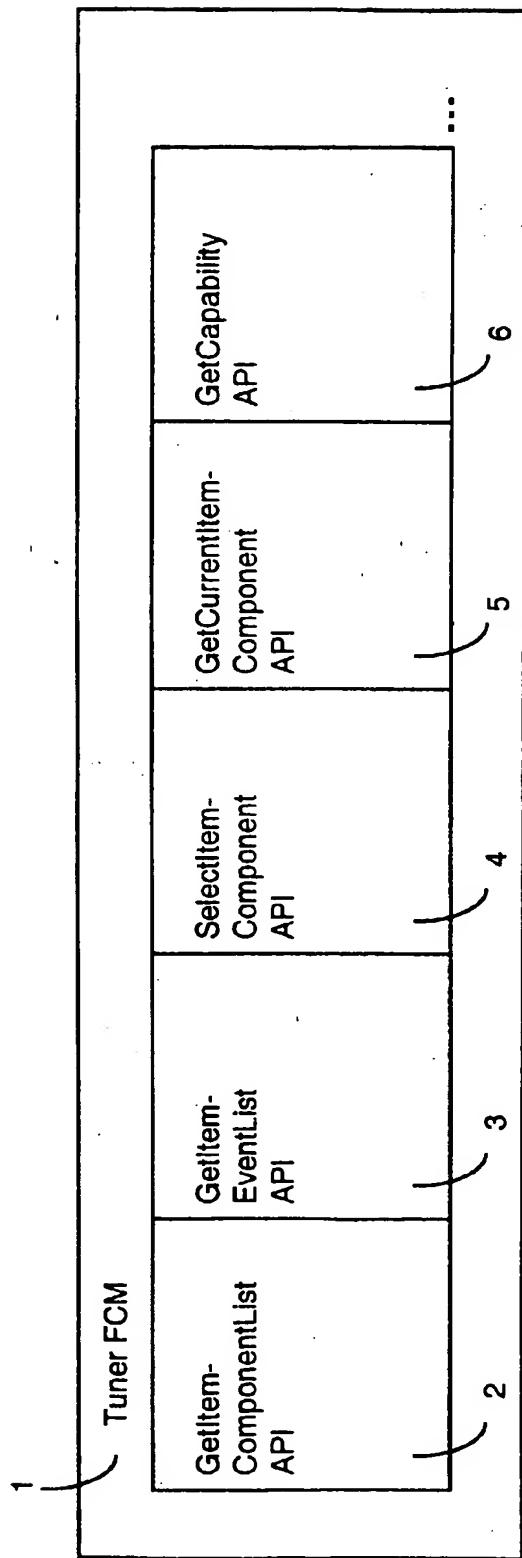


Fig. 1



European Patent  
Office

## EUROPEAN SEARCH REPORT

Application Number

EP 99 10 8916

DOCUMENTS CONSIDERED TO BE RELEVANT			CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	
X	DATABASE INTERNET 'Online! HAVi 19 November 1998 (1998-11-19) "The HAVi Specification - Version 1.0.beta" XP002118795 retrieved from the Internet <http://http://www.havi.org/home.html> 11 October 1999 * page 1 - page 22 * * page 214 - page 220 *	1-11	H04N5/00
A	WO 98 16886 A (PHILIPS ELECTRONICS NV ;PHILIPS NORDEN AB (SE)) 23 April 1998 (1998-04-23) * the whole document *	1-11	
A	BANKS D ET AL: "BREAKING OPEN THE SET TOP BOX" PROCEEDINGS OF THE SPIE, vol. 3228, 4 November 1997 (1997-11-04), pages 105-116, XP002064906 * sections 5 and 6 on pages 110-115 *	1-11	
A	DATABASE INTERNET 'Online! 1394 Trade Association - The Multimedia Connection 15 April 1998 (1998-04-15) "AV/C Tuner Model and Command Set - Version 1.0" XP002118833 retrieved from the Internet <http://www.1394ta.org/Technology/Specific ations/AVC_Tuner10.pdf> 14.10.1999 * the whole document *	1-11	H04N H04B G06F
A	EP 0 893 913 A (CANAL PLUS SA) 27 January 1999 (1999-01-27) * the whole document *	1	
		-/-	
The present search report has been drawn up for all claims			
Place of search	Date of completion of the search	Examiner	
THE HAGUE	14 October 1999	Giannotti, P	
CATEGORY OF CITED DOCUMENTS			
X : particularly relevant if taken alone	T : theory or principle underlying the invention		
Y : particularly relevant if combined with another document of the same category	E : earlier patent document, but published on, or after the filing date		
A : technological background	D : document cited in the application		
O : non-written disclosure	L : document cited for other reasons		
P : intermediate document	& : member of the same patent family, corresponding document		



European Patent  
Office

## EUROPEAN SEARCH REPORT

Application Number  
EP 99 10 8916

DOCUMENTS CONSIDERED TO BE RELEVANT									
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)						
A	EP 0 908 821 A (CANAL PLUS SA) 14 April 1999 (1999-04-14) * abstract * * paragraph '0034! *	1							
A	EVAIN J -P: "THE MULTIMEDIA HOME PLATFORM" EBU REVIEW- TECHNICAL, no. 275, 21 March 1998 (1998-03-21), pages 4-10, XP000767493 ISSN: 0251-0936								
A	ITOH J -I ET AL: "A NEW SOFTWARE ARCHITECTURE FOR EVOLVABLE MULTIMEDIA SOFTWARE" EUROPEAN TRANSACTIONS ON TELECOMMUNICATIONS, vol. 8, no. 4, 1 July 1997 (1997-07-01), pages 423-435, XP000695449 ISSN: 1120-3862								
			TECHNICAL FIELDS SEARCHED (Int.Cl.7)						
<p>The present search report has been drawn up for all claims</p> <table border="1"> <tr> <td>Place of search</td> <td>Date of completion of the search</td> <td>Examiner</td> </tr> <tr> <td>THE HAGUE</td> <td>14 October 1999</td> <td>Giannotti, P</td> </tr> </table> <p>CATEGORY OF CITED DOCUMENTS</p> <p>X : particularly relevant if taken alone  Y : particularly relevant if combined with another document of the same category  A : technological background  O : non-written disclosure  P : intermediate document</p> <p>T : theory or principle underlying the invention  E : earlier patent document, but published on, or after the filing date  D : document cited in the application  L : document cited for other reasons  &amp; : member of the same patent family, corresponding document</p>				Place of search	Date of completion of the search	Examiner	THE HAGUE	14 October 1999	Giannotti, P
Place of search	Date of completion of the search	Examiner							
THE HAGUE	14 October 1999	Giannotti, P							

**ANNEX TO THE EUROPEAN SEARCH REPORT  
ON EUROPEAN PATENT APPLICATION NO.**

EP 99 10 8916

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.  
The members are as contained in the European Patent Office EDP file on  
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

14-10-1999

Patent document cited in search report		Publication date		Patent family member(s)		Publication date
WO 9816886	A	23-04-1998	CN EP	1210601 A 0867004 A	10-03-1999 30-09-1998	
EP 0893913	A	27-01-1999	AU WO	8353498 A 9905854 A	16-02-1999 04-02-1999	
EP 0908821	A	14-04-1999	AU WO	9363298 A 9918730 A	27-04-1999 15-04-1999	